



# Using Implicit Calls to Improve Malware Dynamic Execution

Mourad Leslous, Jean-François Lalande, Valérie Viet Triem Tong

## ► To cite this version:

Mourad Leslous, Jean-François Lalande, Valérie Viet Triem Tong. Using Implicit Calls to Improve Malware Dynamic Execution. 37th IEEE Symposium on Security and Privacy, May 2016, San Jose, United States. hal-01304326

**HAL Id: hal-01304326**

**<https://hal.science/hal-01304326>**

Submitted on 19 May 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Using Implicit Calls to Improve Malware Dynamic Execution

## Dynamic Analysis: The Need for Malware Triggering

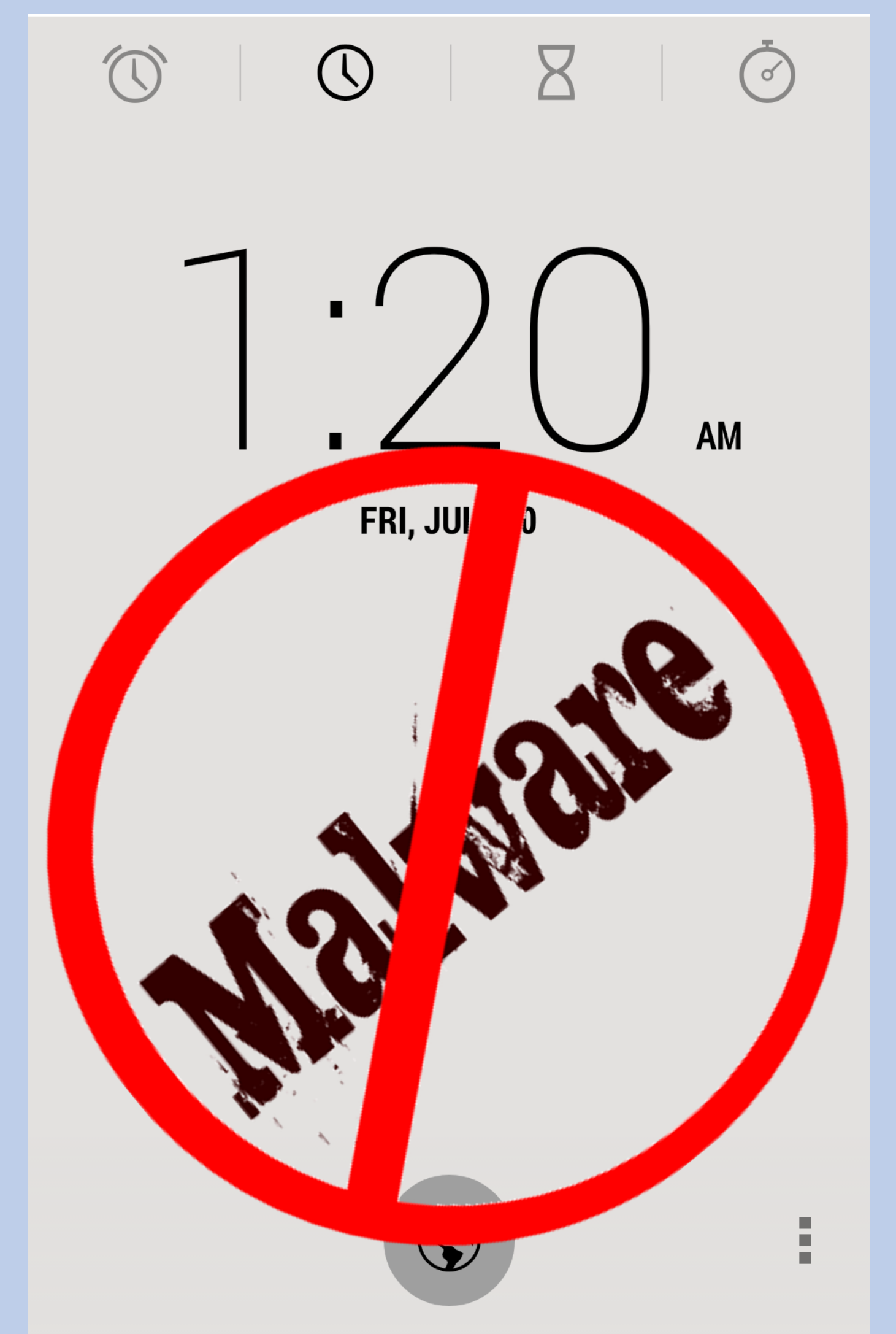
- ▶ Malware evade dynamic analysis
- ▶ Wait for: a specific time, an event, C&C command
- ▶ If **suspicious** code is triggered  
→ Capture dynamic behavior

## GroddDroid: a Gorilla for Triggering Malicious Behaviors

- ▶ Locates the **suspicious** code
- ▶ Builds an interprocedural control flow graph
- ▶ Finds a path that reaches the suspicious code
- ▶ Instruments & Executes APK

## Problem: Execution paths May Use Implicit Calls

- ▶ Alarm Controller (com.al.alarm.controller)
- ▶ SHA256: 03c32aca9f894b8a7263e65fd8845a53618f1131877ccb818d018e2d07e2f596
- ▶ Decrypts and loads code dynamically
- ▶ Installs malicious apps
- ▶ Displays unwanted ads
- ▶ Malicious code execution path contains an implicit call:  
`AsyncTask.execute()` → `AsyncTask.doInBackground()`
- ▶ GroddDroid cannot find such a path because of implicit calls

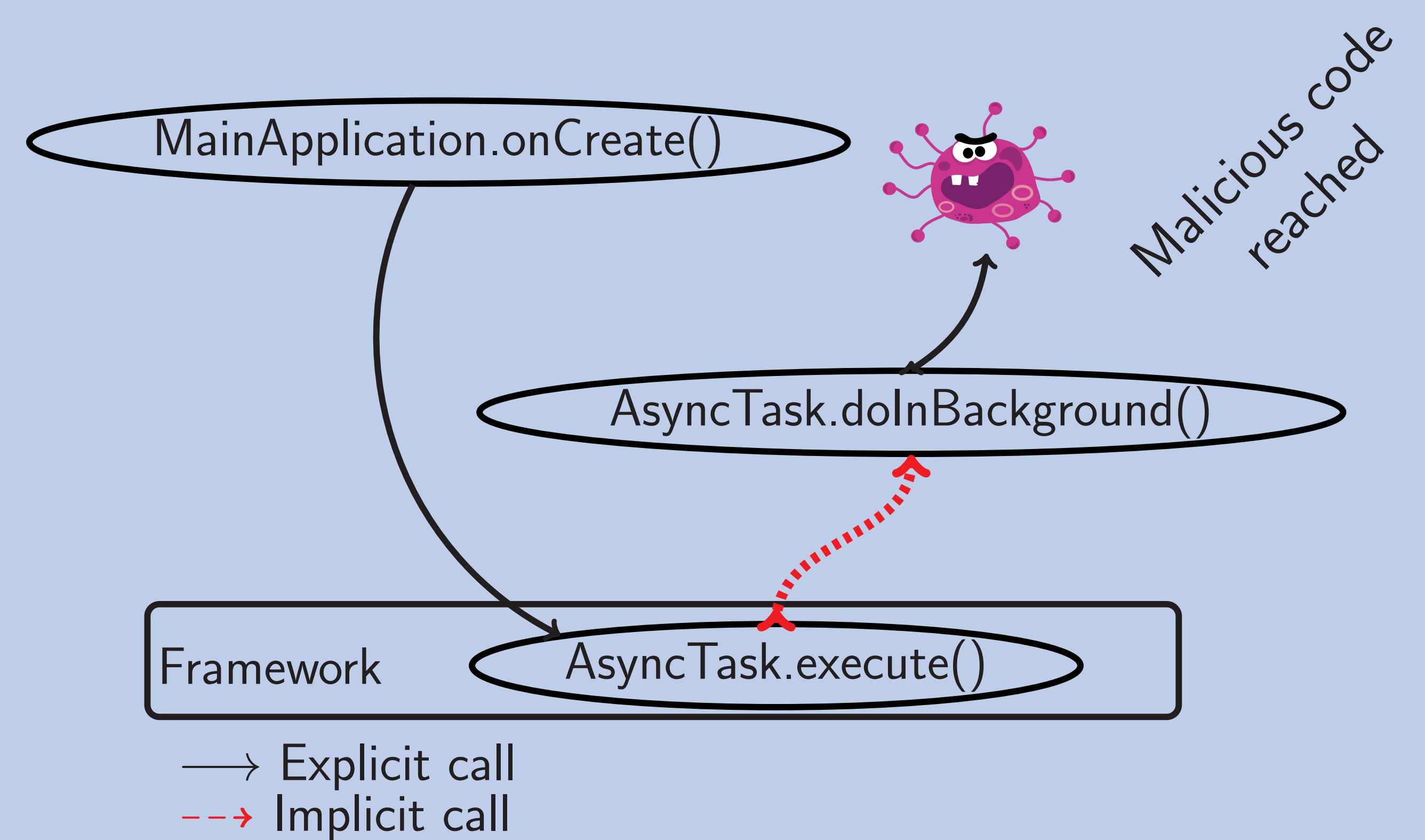
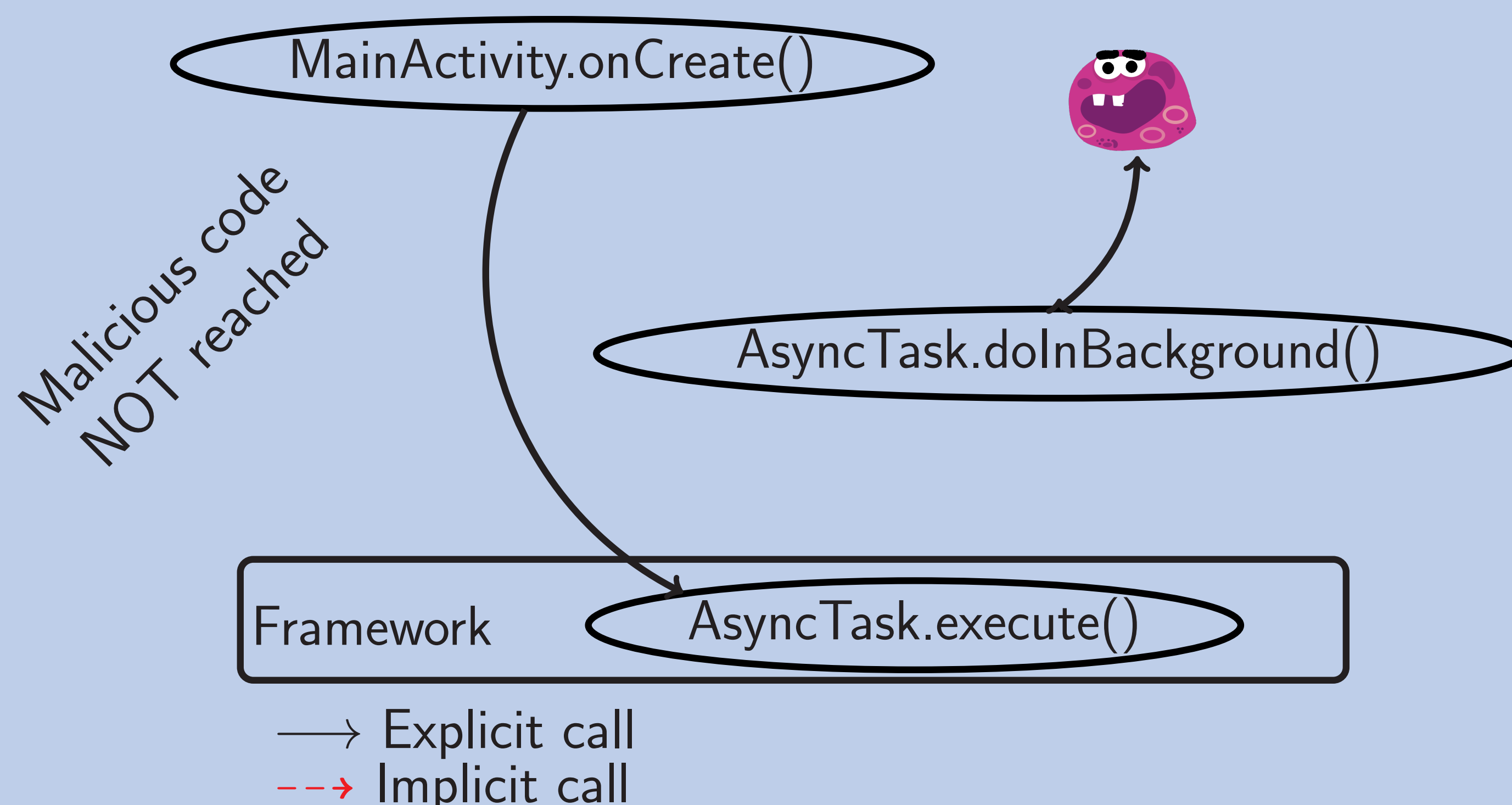


## Challenge

Discover execution paths with implicit calls to reach **suspicious** code

## Including Implicit Calls into CFGs

- ▶ Without implicit calls
- ▶ **Suspicious** code execution rate for this sample: 37.5%
- ▶ Use pairs of *registration-callback*
- ▶ **Suspicious** code execution rate for this sample: 87.5%



## Executing Malware

- ▶ Program designed and implemented
- ▶ Experiments on large sets of malware
- ▶ Measure malicious code coverage with and without implicit calls
- ▶ Online malware analysis platform